Lecture 4: Reasoning with DL

2-AIN-108 Computational Logic

Martin Baláž, Martin Homola

Department of Applied Informatics
Faculty of Mathematics, Physics and Informatics
Comenius University in Bratislava

16 Oct 2012

Preliminaries

Definition (Negation normal form)

A concept C is in negation normal form (NNF) iff the complement constructor ($\neg$) only occurs in front of atomic concept symbols inside C.

Preliminaries (cont.)

Lemma

For every concept C *there exists* C' *in NNF such that* $C \equiv C'$.

Preliminaries (cont.)

Lemma

For every concept C there exists C' in NNF such that $C \equiv C'$.

Proof.

We can always "push" $\neg$ inwards:

- $\neg(E \sqcap F) \equiv \neg E \sqcup \neg F$
- $\neg(E \sqcup F) \equiv \neg E \sqcap \neg F$
- $\neg\exists R.E \equiv \forall R.\neg E$
- $\neg\forall R.E \equiv \exists R.\neg E$

Since each C of finite length we eventually end up with C' in NNF. By structural induction C and C' are equivalent. □

Preliminaries (cont.)

Definition (nnf($\cdot$))

Given any concept C, we denote by nnf(C) a concept C' in NNF s.t. $C \equiv C'$.

Preliminaries (cont.)

Definition (Finite interpretations)

An interpretation $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is finite iff $\Delta^{\mathcal{I}}$ is a finite set.

Definition (Tree-shaped interpretations)

An interpretation $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is tree-shaped iff (V, E), where $V = \Delta^{\mathcal{I}}$ and $E = \{\langle x, y \rangle \mid (\exists R \in N_{\mathrm{R}}) \langle x, y \rangle \in R^{\mathcal{I}}\}$, is a tree.

Preliminaries (cont.)

Definition (Finite model property)

A DL $\mathcal{L}$ is said to have finite model property iff for every satisfiable concept C that can be constructed in $\mathcal{L}$ there exists a finite interpretation $\mathcal{I}$ s.t. $C^{\mathcal{I}} \neq \emptyset$.

Definition (Tree model property)

A DL $\mathcal{L}$ is said to have tree model property iff for every satisfiable concept C that can be constructed in $\mathcal{L}$ there exists a tree-shaped interpretation $\mathcal{I}$ s.t. $C^{\mathcal{I}} \neq \emptyset$.

Definition (Finite tree model property)

A DL $\mathcal{L}$ is said to have finite tree model property iff for every satisfiable concept C that can be constructed in $\mathcal{L}$ there exists a finite tree-shaped interpretation $\mathcal{I}$ s.t. $C^{\mathcal{I}} \neq \emptyset$.

Preliminaries (cont.)

Theorem

$\mathcal{ALC}$ has the finite tree model property.

Corollary

$\mathcal{ALC}$ has the finite model property and the tree model property.

Definition (Completion tree)

A completion tree (CTree) is a triple $T = (V, E, \mathcal{L})$ where (V, E) is a tree and $\mathcal{L}$ is a labeling function s.t.

- $\mathcal{L}(x)$ is a set of concepts for all $x \in V$;
- $\mathcal{L}(\langle x, y \rangle)$ is a set of roles for all $\langle x, y \rangle \in E$.

Tableau Algorithm for $\mathcal{ALC}$ (cont.)

Definition (Completion tree)

A completion tree (CTree) is a triple $T = (V, E, \mathcal{L})$ where (V, E) is a tree and $\mathcal{L}$ is a labeling function s.t.

- $\mathcal{L}(x)$ is a set of concepts for all $x \in V$;
- $\mathcal{L}(\langle x, y \rangle)$ is a set of roles for all $\langle x, y \rangle \in E$.

Definition (Successor, R-successor)

Given a CTree $T = (V, E, \mathcal{L})$ and $x, y \in V$ we say that:

- y is a successor of x iff $\langle x, y \rangle \in E$;
- y is an R-successor of x iff $\langle x, y \rangle \in E$ and $R \in \mathcal{L}(\langle x, y \rangle)$.

Tableau Algorithm for $\mathcal{ALC}$ (cont.)

Definition (Completion tree)

A completion tree (CTree) is a triple $T = (V, E, \mathcal{L})$ where (V, E) is a tree and $\mathcal{L}$ is a labeling function s.t.

- $\mathcal{L}(x)$ is a set of concepts for all $x \in V$;
- $\mathcal{L}(\langle x, y \rangle)$ is a set of roles for all $\langle x, y \rangle \in E$.

Definition (Successor, R-successor)

Given a CTree $T = (V, E, \mathcal{L})$ and $x, y \in V$ we say that:

- y is a successor of x iff $\langle x, y \rangle \in E$;
- y is an R-successor of x iff $\langle x, y \rangle \in E$ and $R \in \mathcal{L}(\langle x, y \rangle)$.

Note: CTrees are representations of interpretations: V corresponds to $\Delta^{\mathcal{I}}$; $\mathcal{L}(x)$ are the concepts to which x belongs; and similarly for $\mathcal{L}(\langle x, y \rangle)$ and $\langle x, y \rangle$.

Tableau Algorithm for $\mathcal{ALC}$ (cont.)

Definition (Clash)

There is a clash in a CTree $T = (V, E, \mathcal{L})$ iff for some $x \in V$ and for some concept C both $C \in \mathcal{L}(x)$ and $\neg C \in \mathcal{L}(x)$.

Tableau Algorithm for $\mathcal{ALC}$ (cont.)

Definition (Clash)

There is a clash in a CTree $T = (V, E, \mathcal{L})$ iff for some $x \in V$ and for some concept C both $C \in \mathcal{L}(x)$ and $\neg C \in \mathcal{L}(x)$.

Definition (Clash-free CTree)

A CTree $T = (V, E, \mathcal{L})$ is clash-free iff there if none of the nodes in V contains a clash.

Tableau Algorithm for $\mathcal{ALC}$ (cont.)

Algorithm (Concept satisfiability)

Input: *concept C in NNF*

Output: *answers if C is satisfiable or not*

Steps:

1. *Initialize a new CTree $T := (\{s_0\}, \emptyset, \{s_0 \mapsto \{C\}\})$;*
2. *Apply tableau expansion rules (next slide) while at least one rule is applicable;*
3. *Answer "C is satisfiable" if T is clash-free. Otherwise answer "C is unsatisfiable".*

Tableau Algorithm for $\mathcal{ALC}$ (cont.)

$\mathcal{ALC}$ tableau expansion rules:

$\sqcap$-rule: if $C_1 \sqcap C_2 \in \mathcal{L}(x)$, $x \in V$ and $\{C_1, C_2\} \not\subseteq \mathcal{L}(x)$
then $\mathcal{L}(x) := \mathcal{L}(x) \cup \{C_1, C_2\}$

$\sqcup$-rule: if $C_1 \sqcup C_2 \in \mathcal{L}(x)$, $x \in V$ and $\{C_1, C_2\} \cap \mathcal{L}(X) = \emptyset$
then either $\mathcal{L}(x) := \mathcal{L}(x) \cup \{C_1\}$ or $\mathcal{L}(x) := \mathcal{L}(x) \cup \{C_2\}$

$\forall$-rule: if $\forall R.C \in \mathcal{L}(x)$, $x, y \in V$, y R-successor of x, $C \notin \mathcal{L}(y)$
then $\mathcal{L}(y) := \mathcal{L}(y) \cup \{C\}$

$\exists$-rule: if $\exists R.C \in \mathcal{L}(x)$, $x \in V$ with no R-successor y s.t. $C \in \mathcal{L}(y)$
then $V := V \cup \{z\}$, $\mathcal{L}(z) := \{C\}$ and $\mathcal{L}(\langle x, z\rangle) := \{R\}$

Tableau Algorithm for $\mathcal{ALC}$ (cont.)

$\mathcal{ALC}$ tableau expansion rules:

$\sqcap$-rule: if $C_1 \sqcap C_2 \in \mathcal{L}(x)$, $x \in V$ and $\{C_1, C_2\} \not\subseteq \mathcal{L}(x)$
then $\mathcal{L}(x) := \mathcal{L}(x) \cup \{C_1, C_2\}$

$\sqcup$-rule: if $C_1 \sqcup C_2 \in \mathcal{L}(x)$, $x \in V$ and $\{C_1, C_2\} \cap \mathcal{L}(X) = \emptyset$
then either $\mathcal{L}(x) := \mathcal{L}(x) \cup \{C_1\}$ or $\mathcal{L}(x) := \mathcal{L}(x) \cup \{C_2\}$

$\forall$-rule: if $\forall R.C \in \mathcal{L}(x)$, $x, y \in V$, y R-successor of x, $C \notin \mathcal{L}(y)$
then $\mathcal{L}(y) := \mathcal{L}(y) \cup \{C\}$

$\exists$-rule: if $\exists R.C \in \mathcal{L}(x)$, $x \in V$ with no R-successor y s.t. $C \in \mathcal{L}(y)$
then $V := V \cup \{z\}$, $\mathcal{L}(z) := \{C\}$ and $\mathcal{L}(\langle x, z\rangle) := \{R\}$

Tableau Algorithm for $\mathcal{ALC}$ (cont.)

Theorem (Correctness)

The tableaux algorithm for deciding satisfiability of concepts always terminates and it is sound and complete.

Tableau Algorithm for $\mathcal{ALC}$ (cont.)

Theorem (Correctness)

The tableaux algorithm for deciding satisfiability of concepts always terminates and it is sound and complete.

For proof see:

- *Attributive concept descriptions with complements.* Schmidt-Schauß, M., Smolka, G. Artificial Intelligence 48(1):1–26, 1991
- *Description logics handbook.* Baader, F., et al., Cambridge University Press, 2003
- *Semantic Investigations in Distributed Ontologies.* Homola, M., PhD. thesis, Comenius University, 2010

Reasoning w.r.t. a TBox

Lemma

$C \sqsubseteq D$ *iff* $\top \sqsubseteq \neg C \sqcup D$

Lemma

$C \sqsubseteq D$ *iff* $\top \sqsubseteq \neg C \sqcup D$

Idea:

- To assure $\mathcal{I} \models C \sqsubseteq D$ we may instead assure that $x \in (\neg C \sqcup D)^{\mathcal{I}}$ for every $x \in \Delta$

Lemma

$C \sqsubseteq D$ *iff* $\top \sqsubseteq \neg C \sqcup D$

Idea:

- To assure $\mathcal{I} \models C \sqsubseteq D$ we may instead assure that $x \in (\neg C \sqcup D)^{\mathcal{I}}$ for every $x \in \Delta$
- Add $\mathrm{nnf}(\neg C \sqcup D)$ to $\mathcal{L}(x)$ for every $x \in V$

Reasoning w.r.t. a TBox

Lemma

$C \sqsubseteq D$ *iff* $\top \sqsubseteq \neg C \sqcup D$

Idea:

- To assure $\mathcal{I} \models C \sqsubseteq D$ we may instead assure that $x \in (\neg C \sqcup D)^{\mathcal{I}}$ for every $x \in \Delta$
- Add $\mathrm{nnf}(\neg C \sqcup D)$ to $\mathcal{L}(x)$ for every $x \in V$

$\mathcal{T}$-rule: if $C_1 \sqsubseteq C_2 \in \mathcal{T}$, $x \in V$ and $\mathrm{nnf}(\neg C_1 \sqcup C_2) \notin \mathcal{L}(x)$
then $\mathcal{L}(x) := \mathcal{L}(x) \cup \{\mathrm{nnf}(\neg C_1 \sqcup C_2)\}$

Reasoning w.r.t. a TBox (cont.)

Problem: naive use of $\mathcal{T}$-rule may lead to infinite looping:

- Let $\mathcal{T} = \{C \sqsubseteq \exists R.C\}$
- Is C satisfiable w.r.t. $\mathcal{T}$?

Reasoning w.r.t. a TBox (cont.)

Definition (Blocking)

Given a CTree $T = (V, E, \mathcal{L})$, a node $x \in V$ is blocked if it has an ancestor y such that

- either $\mathcal{L}(x) \subseteq \mathcal{L}(y)$;
- or y is blocked.

$\mathcal{ALC}$ Tableaux Expansion Rules for TBoxes

$\sqcap$-rule: if $C_1 \sqcap C_2 \in \mathcal{L}(x)$, $x \in V$ and $\{C_1, C_2\} \not\subseteq \mathcal{L}(x)$
and x is not blocked
then $\mathcal{L}(x) := \mathcal{L}(x) \cup \{C_1, C_2\}$

$\sqcup$-rule: if $C_1 \sqcup C_2 \in \mathcal{L}(x)$, $x \in V$ and $\{C_1, C_2\} \cap \mathcal{L}(X) = \emptyset$
and x is not blocked
then either $\mathcal{L}(x) := \mathcal{L}(x) \cup \{C_1\}$ or $\mathcal{L}(x) := \mathcal{L}(x) \cup \{C_2\}$

$\forall$-rule: if $\forall R.C \in \mathcal{L}(x)$, $x, y \in V$, y R-successor of x, $C \notin \mathcal{L}(y)$
and x is not blocked
then $\mathcal{L}(y) := \mathcal{L}(y) \cup \{C\}$

$\exists$-rule: if $\exists R.C \in \mathcal{L}(x)$, $x \in V$ with no R-successor y s.t. $C \in \mathcal{L}(y)$
and x is not blocked
then $V := V \cup \{z\}$, $\mathcal{L}(z) := \{C\}$ and $\mathcal{L}(\langle x, z\rangle) := \{R\}$

$\mathcal{T}$-rule: if $C_1 \sqsubseteq C_2 \in \mathcal{T}$, $x \in V$ and $\mathrm{nnf}(\neg C_1 \sqcup C_2) \notin \mathcal{L}(x)$
and x is not blocked
then $\mathcal{L}(x) := \mathcal{L}(x) \cup \{\mathrm{nnf}(\neg C_1 \sqcup C_2)\}$

Reasoning w.r.t. a TBox (cont.)

Algorithm (Concept satisfiability w.r.t. TBox)

Input: concept C and $\mathcal{T}$ in NNF

Output: answers if C is satisfiable w.r.t. $\mathcal{T}$ or not

Steps:

1. Initialize a new CTree $T := (\{s_0\}, \emptyset, \{s_0 \mapsto \{C\}\})$;
2. Apply tableau expansion rules for TBoxes while at least one rule is applicable;
3. Answer "C is satisfiable w.r.t. $\mathcal{T}$" if T is clash-free. Otherwise answer "C is unsatisfiable w.r.t. $\mathcal{T}$".

Reasoning w.r.t. a TBox (cont.)

Theorem (Correctness)

The tableaux algorithm for deciding satisfiability of concepts w.r.t. a TBox always terminates and it is sound and complete.

Reasoning w.r.t. TBox and ABox

Idea: Encode $\mathcal{A}$ into the CTree

- If $a : C \in \mathcal{A}$
  - $a^{\mathcal{I}} \in C^{\mathcal{I}}$ in every model $\mathcal{I}$
  - add node a into T
  - add C into $\mathcal{L}(a)$
- If $a, b : R \in \mathcal{A}$
  - $\langle a^{\mathcal{I}}, b^{\mathcal{I}} \rangle \in R^{\mathcal{I}}$ in every model $\mathcal{I}$
  - add nodes a, b into T
  - add R into $\mathcal{L}(\langle a, b \rangle)$

Reasoning w.r.t. TBox and ABox

Idea: Encode $\mathcal{A}$ into the CTree

- If $a : C \in \mathcal{A}$
  - $a^\mathcal{I} \in C^\mathcal{I}$ in every model $\mathcal{I}$
  - add node a into T
  - add C into $\mathcal{L}(a)$
- If $a, b : R \in \mathcal{A}$
  - $\langle a^\mathcal{I}, b^\mathcal{I} \rangle \in R^\mathcal{I}$ in every model $\mathcal{I}$
  - add nodes a, b into T
  - add R into $\mathcal{L}(\langle a, b \rangle)$

Note: T is no longer necessarily a tree

Reasoning w.r.t. TBox and ABox

Algorithm (Concept satisfiability w.r.t. TBox and ABox)

Input: *concept C and $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ in NNF*

Output: *answers if C is satisfiable w.r.t. $\mathcal{K}$ or not*

Steps:

1. *Initialize a CTree T as follows:*
   1. $V := \{a \mid \text{constant } a \text{ occurs in } \mathcal{A}\} \cup \{s_0\}$;
   2. $E := \{\langle a, b\rangle \mid a, b : R \in \mathcal{A} \text{ for some role } R\}$;
   3. $\mathcal{L}(a) := \{\mathrm{nnf}(E) \mid a : E \in \mathcal{A}\}$ for all $a \in V$;
      $\mathcal{L}(\langle a, b\rangle) := \{R \mid a, b : R \in \mathcal{A}\}$ for all $\langle a, b\rangle \in E$;
      $\mathcal{L}(s_0) := \{C\}$
2. *Apply tableau expansion rules for TBoxes while at least one rule is applicable;*
3. *Answer "C is satisfiable w.r.t. $\mathcal{K}$" if T is clash-free. Otherwise answer "C is unsatisfiable w.r.t. $\mathcal{K}$".*

Algorithm (Concept satisfiability w.r.t. TBox and ABox)

Input: concept C and $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ in NNF

Output: answers if C is satisfiable w.r.t. $\mathcal{K}$ or not

Steps:

1. *Initialize a CTree T as follows:*
   1. $V := \{a \mid \text{constant } a \text{ occurs in } \mathcal{A}\} \cup \{s_0\}$;
   2. $E := \{\langle a, b\rangle \mid a, b : R \in \mathcal{A} \text{ for some role } R\}$;
   3. $\mathcal{L}(a) := \{\text{nnf}(E) \mid a : E \in \mathcal{A}\}$ for all $a \in V$;
      $\mathcal{L}(\langle a, b\rangle) := \{R \mid a, b : R \in \mathcal{A}\}$ for all $\langle a, b\rangle \in E$;
      $\mathcal{L}(s_0) := \{C\}$
2. *Apply tableau expansion rules for TBoxes while at least one rule is applicable;*
3. *Answer "C is satisfiable w.r.t. $\mathcal{K}$" if T is clash-free. Otherwise answer "C is unsatisfiable w.r.t. $\mathcal{K}$".*

Note: Same algorithm can be used to verify just the consistency of $\mathcal{K}$, simply omit generation of s_0 and its label during the intialization.

Reasoning w.r.t. TBox and ABox

Theorem (Correctness)

The tableaux algorithm for deciding satisfiability of concepts w.r.t. TBox and ABox always terminates and it is sound and complete.